CENTRAL BOARD OF SECONDARY EDUCATION
(COMPUTER SCIENCE – 083)

PRACTICAL FILE

SESSION: 2025–2026

SUBMITTED BY
Name:
Class: XII
Section:
Roll No:

SUBMITTED TO
Teacher's Name:

School Name:
City:

CERTIFICATE

This is to certify that _____of Class XII Section __ has successfully completed the Computer Science Practical File as prescribed by the Central Board of Secondary Education (CBSE) for the academic session _____.

This practical file is the bonafide work of the student and has been completed under my supervision.

Teacher's Signature: _____

Examiner's Signature _____

Date: _____

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my Computer Science teacher for their guidance and support in completing this practical file.

I am also thankful to my school and my parents for their constant encouragement and help throughout the academic session.

INDEX

S. NO	PARTICULARS	PAGE NO
1	Front Page	01
2	Certificate	02
3	Acknowledgement	03
4	Index	04
5	Python Programs	05 - 24
6	SQL Queries	25 - 29
7	Project File	29 - 39

PYTHON PROGRAMS

Program No. 1

Aim:

To read a text file line by line and display each word separated by a #.

Algorithm:

1. Open the text file in read mode.
2. Read the file line by line.
3. Split each line into words.
4. Join the words using the symbol #.
5. Display the output.

Program Code:

```
with open("data.txt", "r") as f:
    for line in f:
        words = line.split()
        print("#".join(words))
```

Console:

```
Input file (data.txt):
This is a test file
```

Output:

```
This#is#a#test#file
```

Program No. 2

Aim:

To read a text file and display the number of vowels, consonants, uppercase and lowercase characters.

Algorithm:

1. Open the text file in read mode.
2. Initialize counters for vowels, consonants, uppercase and lowercase letters.
3. Read the file character by character.
4. For each character:
   a. Check if it is a letter.
   b. If it is a vowel, increment vowel counter; otherwise increment consonant counter.
   c. If it is uppercase, increment uppercase counter.
   d. If it is lowercase, increment lowercase counter.
5. Display all counts.

Program Code:

```
with open("data.txt", "r") as f:
    text = f.read()
    for ch in text:
        if ch.isalpha():
            if ch.lower() in "aeiou":
                vowels += 1
            else:
                consonants += 1
            if ch.isupper():
                uppercase += 1
            if ch.islower():
                lowercase += 1

print("Vowels:", vowels)
print("Consonants:", consonants)
print("Uppercase letters:", uppercase)
print("Lowercase letters:", lowercase)
```

Console:

```
Input file (data.txt):
Hello World
```

Output:

```
Vowels: 3
Consonants: 7
Uppercase letters: 2
Lowercase letters: 8
```

Program No. 3

Aim:

To remove all the lines that contain the character 'a' in a file and write the remaining lines to another file.

Algorithm:

1. Open the input file in read mode.
2. Open the output file in write mode.
3. Read the input file line by line.
4. For each line, check if it contains the character 'a'.
   a. If it does NOT contain 'a', write it to the output file.
5. Close both files.
6. Display a message that the operation is complete.

Program Code:

```
with open("input.txt", "r") as infile, open("output.txt", "w") as
outfile:
    for line in infile:
        if 'a' not in line:
            outfile.write(line)

print("Lines without 'a' have been written to output.txt")
```

Console:

```
Input file (input.txt):
apple
orange
hello
world
```

Output:

```
Output file (output.txt):
hello
world
```

Program No. 4

Aim:

To create a binary file with name and roll number, and search for a given roll number to display the corresponding name.

Algorithm:

1. Import the `pickle` module for binary file handling.
2. Create a dictionary with roll numbers as keys and names as values.
3. Open a binary file in write mode and save the dictionary using `pickle.dump()`.
4. Open the binary file in read mode.
5. Input a roll number to search.
6. Check if the roll number exists in the dictionary:
   a. If found, display the name.
   b. If not found, display an appropriate message.
7. Close the file.

Program Code:

```
import pickle

# Step 1: Create and save data
students = {1: "Krishna", 2: "Aditya", 3: "Rohit"}

with open("students.dat", "wb") as f:
    pickle.dump(students, f)

# Step 2: Search for a roll number
roll = int(input("Enter roll number to search: "))

with open("students.dat", "rb") as f:
    data = pickle.load(f)
    if roll in data:
        print("Name:", data[roll])
    else:
        print("Roll number not found")
```

Console:

```
Enter roll number to search: 2
Name: Aditya
```

Output:

```
Enter roll number to search: 5
Roll number not found
```

Program No. 5

Aim:

To create a binary file with roll number, name, and marks. Update the marks for a given roll number.

Algorithm:

1. Import the `pickle` module for binary file handling.
2. Create a dictionary with roll numbers as keys and a list of [name, marks] as values.
3. Open a binary file in write mode and save the dictionary using `pickle.dump()`.
4. Open the binary file in read mode and load the data.
5. Input the roll number whose marks need to be updated.
6. If the roll number exists:
   a. Input the new marks.
   b. Update the marks in the dictionary.
   c. Save the updated dictionary back to the binary file.
7. If the roll number does not exist, display a message.
8. Display updated data.
9. Close the file.

Program Code:

```
import pickle

# Step 1: Create and save initial data
students = {1: ["Krishna", 85], 2: ["Aditya", 90], 3: ["Rohit", 75]}

with open("marks.dat", "wb") as f:
    pickle.dump(students, f)

# Step 2: Update marks
roll = int(input("Enter roll number to update marks: "))

with open("marks.dat", "rb") as f:
    data = pickle.load(f)

if roll in data:
    new_marks = int(input("Enter new marks: "))
    data[roll][1] = new_marks
    print("Marks updated successfully.")
```

```
else:
    print("Roll number not found.")

# Save updated data back to file
with open("marks.dat", "wb") as f:
    pickle.dump(data, f)

# Display updated data
print("Updated Student Data:")
for r, info in data.items():
    print("Roll:", r, "Name:", info[0], "Marks:", info[1])
```

Console:

```
Enter roll number to update marks: 2
Enter new marks: 95
```

Output:

```
Marks updated successfully.
Updated Student Data:
Roll: 1 Name: Krishna Marks: 85
Roll: 2 Name: Aditya Marks: 95
Roll: 3 Name: Rohit Marks: 75
```

Program No. 6

Aim:

To write a random number generator that generates random numbers between 1 and 6 (simulates a dice)

Algorithm:

1. Import “random” module
2. Print the randint between 0 to 6.

Program Code:

```
import random

print(random.randint(0, 6));
```

Output:

```
Upon executing again and again.
5
2
1
0
6
```

Program No 7.

Aim:

To write a Python program to implement a stack using list

Algorithm:

1. Defining a stack variable as list.
2. Appending elements A, B, C to it using built in list append functions
3. Printing the stack out to the console.
4. Peeking the top element.
5. Popping the list.

Program Code:

```
stack = []

# Push
stack.append("A")
stack.append("B")
stack.append("C")
print("Stack: ", stack)

# Peek
topElement = stack[-1]
print("Peek: ", topElement)

# Pop
poppedElement = stack.pop()
print("Pop: ", poppedElement)

# Stack after Pop
print("Stack after Pop: ", stack)

# isEmpty
isEmpty = not bool(stack)
print("isEmpty: ", isEmpty)

# Size
```

```
print("Size: ", len(stack))
```

Output:

```
Stack:  ['A', 'B', 'C']
Peek:  C
Pop:  C
Stack after Pop:  ['A', 'B']
isEmpty:  False
Size:  2
```

Program No 8.

Aim:

To create a CSV file by entering User ID and Password, and to read and search the password for a given User ID.

Algorithm:

1. Start
2. Import the `csv` module
3. Open a CSV file in write mode
4. Enter User ID and Password from the user
5. Write the entered data into the CSV file
6. Close the file
7. Open the CSV file in read mode
8. Enter the User ID to be searched
9. Read each row from the CSV file
10. If the entered User ID matches, display the corresponding password
11. If no match is found, display “User ID not found”
12. Stop

Program Code:

```
import csv

# Create CSV file and write data
with open("users.csv", "w", newline="") as file:
    writer = csv.writer(file)
    writer.writerow(["UserID", "Password"])

    n = int(input("Enter number of users: "))
    for i in range(n):
        uid = input("Enter User ID: ")
        pwd = input("Enter Password: ")
        writer.writerow([uid, pwd])

# Read and search password
search_id = input("\nEnter User ID to search password: ")
found = False
```

```
with open("users.csv", "r") as file:
    reader = csv.reader(file)
    next(reader)  # skip header

    for row in reader:
        if row[0] == search_id:
            print("Password is:", row[1])
            found = True
            break

if not found:
    print("User ID not found")
```

Output:

```
Enter number of users: 2
Enter User ID: admin
Enter Password: admin123
Enter User ID: user1
Enter Password: pass456

Enter User ID to search password: user1
Password is: pass456
```

Program No 9.

Aim:

To read a text file and write its contents into another file in reverse order.

Algorithm:

1. Start
2. Open the source file in read mode
3. Read all lines from the file
4. Reverse the order of lines
5. Open the destination file in write mode
6. Write the reversed lines into the new file
7. Close both files
8. Stop

Program Code:

```
source = open("source.txt", "r")
lines = source.readlines()
source.close()

dest = open("reverse.txt", "w")
for line in reversed(lines):
    dest.write(line)
dest.close()

print("File copied in reverse order")

## Output
File copied in reverse order
```

Output:

```
File copied in reverse order
```

Program No 10.

Aim:

To count the number of lines, words, and characters in a text file.

Algorithm:

1. Start
2. Open the text file in read mode
3. Read all lines
4. Count number of lines
5. Count words and characters
6. Display the counts
7. Close the file
8. Stop

Program Code:

```
file = open("data.txt", "r")
lines = file.readlines()

line_count = len(lines)
word_count = 0
char_count = 0

for line in lines:
    word_count += len(line.split())
    char_count += len(line)

file.close()

print("Lines:", line_count)
print("Words:", word_count)
print("Characters:", char_count)
```

Output:

```
Lines: 3
Words: 18
Characters: 96
```

Program No 11.

Aim:

To search for a given word in a text file and display the line number.

Algorithm:

1. Start
2. Open the file in read mode
3. Read file line by line
4. Search for the given word
5. Display line number if found
6. If not found, display message
7. Stop

Program Code:

```
word = input("Enter word to search: ")
file = open("data.txt", "r")

found = False
line_no = 1

for line in file:
    if word in line:
        print("Word found at line", line_no)
        found = True
    line_no += 1

if not found:
    print("Word not found")

file.close()
```

Output:

```
Enter word to search: Python
Word found at line 2
```

Program No 12.

Aim:

To create a CSV file of student records and display students scoring more than 75 marks.

Algorithm:

1. Start
2. Create a CSV file
3. Enter roll number, name, and marks
4. Read the CSV file
5. Display students with marks > 75
6. Stop

Program Code:

```
import csv

with open("students.csv", "w", newline="") as file:
    writer = csv.writer(file)
    writer.writerow(["Roll", "Name", "Marks"])
    for i in range(3):
        r = input("Roll: ")
        n = input("Name: ")
        m = int(input("Marks: "))
        writer.writerow([r, n, m])

print("\nStudents scoring above 75:")

with open("students.csv", "r") as file:
    reader = csv.reader(file)
    next(reader)
    for row in reader:
        if int(row[2]) > 75:
            print(row)
```

Output:

```
Students scoring above 75:
['2', 'Rahul', '82']
```

Program No 13.

Aim:

To implement a queue using list in Python.

Algorithm:

1. Start
2. Create an empty list
3. Insert elements using enqueue
4. Remove elements using dequeue
5. Display the queue
6. Stop

Program Code:

```
queue = []

queue.append(10)
queue.append(20)
queue.append(30)

print("Queue:", queue)

queue.pop(0)
print("After Dequeue:", queue)
```

Output:

```
Queue: [10, 20, 30]
After Dequeue: [20, 30]
```

Program No 14.

Aim:

To create a binary file with employee ID and name, and display all records.

Algorithm:

1. Start
2. Create a binary file
3. Write employee records using pickle
4. Read and display records
5. Stop

Program Code:

```
import pickle

file = open("emp.dat", "wb")
data = [
    {"id": 1, "name": "Krishna"},
    {"id": 2, "name": "Aditya"}
]
pickle.dump(data, file)
file.close()

file = open("emp.dat", "rb")
records = pickle.load(file)

for r in records:
    print(r)

file.close()
```

Output:

```
{'id': 1, 'name': 'Krishna'}
{'id': 2, 'name': 'Aditya'}
```

Program No 15.

Aim:

To create a text file and count the frequency of each word present in the file.

Algorithm:

1. Start
2. Open the text file in read mode
3. Read the entire content of the file
4. Split the content into words
5. Count the frequency of each word
6. Display each word with its frequency
7. Close the file
8. Stop

Program Code:

```
file = open("words.txt", "r")
text = file.read().lower()
file.close()

words = text.split()
freq = {}

for word in words:
    if word in freq:
        freq[word] += 1
    else:
        freq[word] = 1

for word in freq:
    print(word, ":", freq[word])
```

Output:

```
python : 3
file : 2
program : 1
```

SQL QUERIES

1. Create a table named store

```
MariaDB [PROJECT]> CREATE TABLE store (
    ->     itemno INT PRIMARY KEY,
    ->     item CHAR(20),
    ->     scode INT,
    ->     qty INT,
    ->     rate INT,
    ->     lastbuy DATE
    -> );
Query OK, 0 rows affected (0.010 sec)
```

2. Display all details of store

```
MariaDB [PROJECT]> SELECT * FROM store;
+--------+-----------+-------+------+------+------------+
| itemno | item      | scode | qty  | rate | lastbuy    |
+--------+-----------+-------+------+------+------------+
|      1 | Sharpener |    23 |  100 |    5 | 2025-01-29 |
|      2 | Ball Pen  |    22 |  200 |   10 | 2025-01-29 |
|      3 | Gel Pen   |    21 |  150 |   15 | 2025-01-29 |
|      4 | Eraser    |    22 |   50 |    3 | 2025-01-29 |
+--------+-----------+-------+------+------+------------+
4 rows in set (0.001 sec)
```

3. Display details of all items in the store table in descending order of item no.

```
MariaDB [PROJECT]> SELECT * FROM store ORDER BY itemno DESC;
+--------+-----------+-------+------+------+------------+
| itemno | item      | scode | qty  | rate | lastbuy    |
+--------+-----------+-------+------+------+------------+
|      4 | Eraser    |    22 |   50 |    3 | 2025-01-29 |
|      3 | Gel Pen   |    21 |  150 |   15 | 2025-01-29 |
|      2 | Ball Pen  |    22 |  200 |   10 | 2025-01-29 |
|      1 | Sharpener |    23 |  100 |    5 | 2025-01-29 |
+--------+-----------+-------+------+------+------------+
4 rows in set (0.000 sec)
```

4. Display item no and item name of those items whose rate is more than 5 rupees

```
MariaDB [PROJECT]> SELECT itemno, item FROM store WHERE rate > 5;
+--------+----------+
| itemno | item     |
+--------+----------+
|      2 | Ball Pen |
|      3 | Gel Pen  |
+--------+----------+
2 rows in set (0.001 sec)
```

5. Display details of those items whose code is 22 or qty is more than 110

```
MariaDB [PROJECT]> SELECT * FROM store WHERE scode = 22 OR qty > 110;
+--------+----------+-------+------+------+------------+
| itemno | item     | scode | qty  | rate | lastbuy    |
+--------+----------+-------+------+------+------------+
|      2 | Ball Pen |    22 |  200 |   10 | 2025-01-29 |
|      3 | Gel Pen  |    21 |  150 |   15 | 2025-01-29 |
|      4 | Eraser   |    22 |   50 |    3 | 2025-01-29 |
+--------+----------+-------+------+------+------------+
3 rows in set (0.001 sec)
```

6. Display item name and rate if 2% discount is given on each item

```
MariaDB [PROJECT]> SELECT item, rate, rate * 0.98 AS discounted_rate FROM store;
+-----------+------+-----------------+
| item      | rate | discounted_rate |
+-----------+------+-----------------+
| Sharpener |    5 |            4.90 |
| Ball Pen  |   10 |            9.80 |
| Gel Pen   |   15 |           14.70 |
| Eraser    |    3 |            2.94 |
+-----------+------+-----------------+
4 rows in set (0.000 sec)
```

7. Display no. of distinct scode from store

```
MariaDB [PROJECT]> SELECT COUNT(DISTINCT scode) AS distinct_scode_count FROM store;
+----------------------+
| distinct_scode_count |
+----------------------+
|                    3 |
+----------------------+
1 row in set (0.001 sec)
```

8. Display maximum lastbuy from store

```
MariaDB [PROJECT]> SELECT MAX(lastbuy) AS latest_purchase_date FROM store;
+----------------------+
| latest_purchase_date |
+----------------------+
| 2025-01-29           |
+----------------------+
1 row in set (0.000 sec)
```

9. Create table suppliers and insert rows according to scode

```
MariaDB [PROJECT]> CREATE TABLE suppliers (
    ->     scode INT PRIMARY KEY,
    ->     sname VARCHAR(100) NOT NULL,
    ->     address VARCHAR(255),
    ->     contact_number VARCHAR(20)
    -> );
Query OK, 0 rows affected (0.009 sec)

MariaDB [PROJECT]> INSERT INTO suppliers (scode, sname, address, contact_number)
    -> VALUES
    -> (101, 'Supplier A', '123 Street, City, Country', '123-456-7890'),
    -> (102, 'Supplier B', '456 Avenue, City, Country', '987-654-3210'),
    -> (103, 'Supplier C', '789 Road, City, Country', '555-123-4567');
Query OK, 3 rows affected (0.007 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

10. Select item no. item name and supplier name where qty is greater than 50

```
MariaDB [PROJECT]> SELECT store.itemno, store.item, suppliers.sname
    -> FROM store
    -> JOIN suppliers ON store.scode = suppliers.scode
    -> WHERE store.qty > 50;
Empty set (0.001 sec)
```

11. Display total quantity group by scode

```
MariaDB [PROJECT]> SELECT scode, SUM(qty) AS total_quantity
    -> FROM store
    -> GROUP BY scode;
+-------+----------------+
| scode | total_quantity |
+-------+----------------+
|    21 |            150 |
|    22 |            250 |
|    23 |            100 |
+-------+----------------+
3 rows in set (0.001 sec)
```

12. display details of those items where r is present in the item name

```
MariaDB [PROJECT]> SELECT * FROM store
    -> WHERE item LIKE '%r%';
+--------+-----------+-------+------+------+------------+
| itemno | item      | scode | qty  | rate | lastbuy    |
+--------+-----------+-------+------+------+------------+
|      1 | Sharpener |    23 |  100 |    5 | 2025-01-29 |
|      4 | Eraser    |    22 |   50 |    3 | 2025-01-29 |
+--------+-----------+-------+------+------+------------+
2 rows in set (0.000 sec)
```

13. Delete column qty from store table

```
MariaDB [PROJECT]> ALTER TABLE store
    -> DROP COLUMN qty;
Query OK, 0 rows affected, 1 warning (0.011 sec)
Records: 0  Duplicates: 0  Warnings: 1
```

14. Change scode of ball pen to 23

```
MariaDB [PROJECT]> UPDATE store
    -> SET scode = 23
    -> WHERE item = 'Ball Pen';
Query OK, 1 row affected (0.007 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

15. Delete table store

```
MariaDB [PROJECT]> DROP TABLE store;
Query OK, 0 rows affected (0.009 sec)
```

Project File

```
import mysql.connector
import time

db = mysql.connector.connect(
      host="localhost",
      user="root",
      password="1234",
      database="contact"
)

cursor = db.cursor()
cursor.execute("""
CREATE TABLE IF NOT EXISTS book (
      name char(30) primary key,
      address char(100),
      mobile char(15),
      email char(30)
);
""")

def intro():
      print("=" * 80)
      print("{:^80s}".format("CONTACT"))
      print("{:^80s}".format("BOOK"))
      print("{:^80s}".format("PROJECT"))
      print("{:^80s}".format("MADE BY: Student"))
      print("=" * 80)
      print()
      time.sleep(2)

def create_record():
      name = input("Enter name: ")
      address = input("Enter address: ")
      mobile = input("Enter mobile: ")
      email = input("Enter email: ")
      sql = "INSERT INTO book(name,address,mobile,email) VALUES
(%s,%s,%s,%s)"
      record = (name, address, mobile, email)
      cursor.execute(sql, record)
      db.commit()
      print("Record Entered Successfully\n")
```

```
def search(name):
      sql = "SELECT * FROM book WHERE name = %s"
      value = (name,)
      cursor.execute(sql, value)
      record = cursor.fetchone()
      if record is None:
      print("No such record exists")
      else:
      print('Name:', record[0])
      print('Address:', record[1])
      print('Mobile:', record[2])
      print('E-mail:', record[3])

def display_all():
      cursor.execute("SELECT * FROM book")
      print('{0:20}{1:30}{2:15}{3:30}'.format('NAME', 'ADDRESS',
'MOBILE NO', 'E-MAIL'))
      for record in cursor:
      print('{0:20}{1:30}{2:15}{3:30}'.format(record[0], record[1],
record[2], record[3]))

def delete_record(name):
      sql = "DELETE FROM book WHERE name = %s"
      value = (name,)
      cursor.execute(sql, value)
      db.commit()
      if cursor.rowcount == 0:
      print("Record not found")
      else:
      print("Record deleted successfully")

def modify_record(name):
      sql = "SELECT * FROM book WHERE name = %s"
      value = (name,)
      cursor.execute(sql, value)
      record = cursor.fetchone()
      if record is None:
      print("No such record exists")
      else:
      while True:
            print("\nPress the option you want to edit: ")
            print("1. Name")
            print("2. Address")
            print("3. Mobile")
```

```
            print("4. BACK")
            print()
            ch = int(input("Select Your Option (1-4): "))
            if ch == 1:
                  new_name = input("Enter new name: ")
                  sql = "UPDATE book SET name = %s WHERE name = %s"
                  values = (new_name, name)
                  cursor.execute(sql, values)
                  db.commit()
                  print(cursor.rowcount, "record updated successfully")
            elif ch == 2:
                  new_address = input("Enter new address: ")
                  sql = "UPDATE book SET address = %s WHERE name = %s"
                  values = (new_address, name)
                  cursor.execute(sql, values)
                  db.commit()
                  print(cursor.rowcount, "record updated successfully")
            elif ch == 3:
                  new_mobile = input("Enter new mobile : ")
                  sql = "UPDATE book SET mobile = %s WHERE name = %s"
                  values = (new_mobile, name)
                  cursor.execute(sql, values)
                  db.commit()
                  print(cursor.rowcount, "record updated successfully")
            elif ch == 4:
                  break
            else:
                  print("Invalid choice !!!\n")

def main():
      intro()
      while True:
      print("\nMAIN MENU")
      print("1. ADD NEW RECORD")
      print("2. SEARCH RECORD")
      print("3. DISPLAY ALL RECORDS")
      print("4. DELETE RECORD")
      print("5. MODIFY RECORD")
      print("6. EXIT")
      print()
      ch = int(input("Select Your Option (1-6): "))
      print()
      if ch == 1:
            print("ADD NEW RECORD")
```

```
            create_record()
        elif ch == 2:
            print("SEARCH RECORD BY NAME")
            name = input("Enter name: ")
            search(name)
        elif ch == 3:
            print("DISPLAY ALL RECORDS")
            display_all()
        elif ch == 4:
            print("DELETE RECORD")
            name = input("Enter name: ")
            delete_record(name)
        elif ch == 5:
            print("MODIFY RECORD")
            name = input("Enter name: ")
            modify_record(name)
        elif ch == 6:
            print("Thanks for using Contact Book")
            db.close()
            break
        else:
            print("Invalid choice")

main()
```

Output

Adding new record:

```
MAIN MENU
1. ADD NEW RECORD
2. SEARCH RECORD
3. DISPLAY ALL RECORDS
4. DELETE RECORD
5. MODIFY RECORD
6. EXIT

Select Your Option (1-6): 1

ADD NEW RECORD
Enter name: Abc
Enter address: 2nd-Street
Enter mobile: 9876543210
Enter email: abc@gmail.com
Record Entered Successfully
```

Search record by name:

```
Select Your Option (1-6): 2

SEARCH RECORD BY NAME
Enter name: abc
Name: Abc
Address: 2nd-Street
Mobile: 9876543210
E-mail: abc@gmail.com
```

If record not found:

```
SEARCH RECORD BY NAME
Enter name: test
No such record exists
```

Display all records:

```
MAIN MENU
1. ADD NEW RECORD
2. SEARCH RECORD
3. DISPLAY ALL RECORDS
4. DELETE RECORD
5. MODIFY RECORD
6. EXIT


Select Your Option (1-6): 3

DISPLAY ALL RECORDS
NAME                ADDRESS             MOBILE NO           E-MAIL
Abc                 2nd-Street          9876543210          abc@gmail.com
Def                 Oak hill            8642013579          def99@gmail.com
Pqr                 Las Brujas          6138401474          pqr.z1@gmail.com
```

Edit records:

```
MAIN MENU
1. ADD NEW RECORD
2. SEARCH RECORD
3. DISPLAY ALL RECORDS
4. DELETE RECORD
5. MODIFY RECORD
6. EXIT

Select Your Option (1-6): 5

MODIFY RECORD
Enter name: pqr

Press the option you want to edit:
1. Name
2. Address
3. Mobile
4. BACK

Select Your Option (1-4): 3
Enter new mobile: 9753124680
1 record updated successfully
```

Delete records:

```
MAIN MENU
1. ADD NEW RECORD
2. SEARCH RECORD
3. DISPLAY ALL RECORDS
4. DELETE RECORD
5. MODIFY RECORD
6. EXIT

Select Your Option (1-6): 4
DELETE RECORD
Enter name: def
Record deleted Successfully
```

Exiting program

```
MAIN MENU
1. ADD NEW RECORD
2. SEARCH RECORD
3. DISPLAY ALL RECORDS
4. DELETE RECORD
5. MODIFY RECORD
6. EXIT

Select Your Option (1-6): 6

Thanks for using Contact Book
```